

## README for CSE 464

### Project Part 3

Parth Shah

1225457038

GitHub Repository link: <https://github.com/Parth576/CSE-464-2023-prshah11>

#### 1. Creating a refactor branch and perform 5 refactor commits

- The first refactor was to encapsulate the 'nodes' variable to follow the OOP paradigm of encapsulation, that is, make the 'nodes' variable private and expose a getter method 'getNodeList'. The commit link for this is:  
<https://github.com/Parth576/CSE-464-2023-prshah11/commit/98b7e214ae99e6c7a9a6549477a41ea1f586434c>
- The second refactor improves the error handling in the string functions 'toString' and 'constructOutputString'. Since the 'toString' function returns a string we return the error in a custom string as well as throw an exception. The link to this commit is:  
<https://github.com/Parth576/CSE-464-2023-prshah11/commit/1ee23f2eca9ff7dbc5c2c71d2d4eba10327e91>
- The third refactor commit was renaming some function names to make their usage more apparent. I used a bit more descriptive names for the functions like export/import followed by the file extension. For example, 'parseGraph' was changed to 'importGraphFromDot' and 'outputGraphics' was replaced with 'exportGraphToPNG'. The link for this commit:  
<https://github.com/Parth576/CSE-464-2023-prshah11/commit/f111b0e5df95cedc6dd928f837f0dfcd97cb043c>
- For the fourth refactor commit, I refactored the code within the 'containsNode' method to use newer and efficient java syntax for searching within an array. This refactor made the code much shorter. The link to this commit:  
<https://github.com/Parth576/CSE-464-2023-prshah11/commit/75430bbb0400a6343e199fd9d669674d2b250c68>
- Before, I was using a boolean variable to keep track if any of the nodes failed to be removed and broke out from the loop. This was very hacky so I refactored the 'removeNodes' method and maintained a list of failed nodes which I am printing at the end. Now the function is much cleaner and easier to understand as well. The link to this commit:  
<https://github.com/Parth576/CSE-464-2023-prshah11/commit/dbaba160f386084b1327c908ee774b695f9ad79b>
- This was just an extra refactoring commit that added a getter method 'getGraph' for the graph variable. This was a preparatory commit for applying the design patterns. The link to this commit is:  
<https://github.com/Parth576/CSE-464-2023-prshah11/commit/ca2c016eba1c1efefc6f7b5475e86d88c9dd02e3>

## 2. Applying template design pattern for BFS and DFS

- First, I added a 'GraphSearchTemplate' class which had all the methods common to the search algorithms. Only the 'search' method was left abstract and will be implemented separately in the search algorithms. The commit link for this is: <https://github.com/Parth576/CSE-464-2023-prshah11/commit/5b12600e0008dde1ab3e073ec4f5f8be2162866f>
- Next, I created the separate classes for BFS and DFS which inherited the functions from the GraphSearchTemplate and then I implemented the separate search methods for BFS and DFS. Commit link: <https://github.com/Parth576/CSE-464-2023-prshah11/commit/f57acbcfb8fd78e25b7c25ff9637c0b1a09996a4>
- I added tests for testing BFS and DFS using the template design pattern. The commit link: <https://github.com/Parth576/CSE-464-2023-prshah11/commit/d86b96b85867ac6a16be3185e770280418ab1ce7>
- Example usage with template pattern:

```
GraphManager gm = new GraphManager();
gm.importGraphFromDOT("src/test2.dot");
Graph<String,DefaultEdge> currGraph = gm.getGraph()
BreadthFirstSearch bfs = new BreadthFirstSearch(currGraph);
GraphManager.Path result = bfs.search("a", "h");
```

## 3. Applying strategy pattern for BFS and DFS

- Added a class 'GraphSearchContext' and an interface 'SearchStrategy' as the first step to refactoring according to strategy pattern. Commit link: <https://github.com/Parth576/CSE-464-2023-prshah11/commit/74467edb30de80daf2491b679d88c1fcf8107b8a>
- Make the BFS and DFS classes (created while refactoring according to template pattern) implement the SearchStrategy interface as well. Commit link: <https://github.com/Parth576/CSE-464-2023-prshah11/commit/74467edb30de80daf2491b679d88c1fcf8107b8a>
- Rewrote the 'GraphSearch' API which was created in part 2 of the project to now use the strategy pattern. The code looks much cleaner and understandable now. It was easier to debug as well. Commit link: <https://github.com/Parth576/CSE-464-2023-prshah11/commit/2bdca0f95aca6895e8d2913856dcece441d1f58>
- Renamed the tests so that we can see the difference between template pattern and strategy pattern. Did not need to rewrite the test as the API is the same, just the internal code has changed. <https://github.com/Parth576/CSE-464-2023-prshah11/commit/a21818bf8481bf3c8800661ef275fe2b9aa9b70d>
- Example usage with strategy pattern :

```
GraphManager gm = new GraphManager();  
gm.importGraphFromDOT("src/test2.dot");  
GraphManager.Path result = gm.GraphSearch("a", "h", GraphManager.Algorithm.DFS);
```

#### 4. Implementing random walk algorithm

- Add an additional option to print the visited path while running the search algorithm in the 'constructPath' method.  
<https://github.com/Parth576/CSE-464-2023-prshah11/commit/6d660b04ee2e5080c635ff52a1567e01a70abcf3>
- Added the random walk algorithm using the 'RandomWalkIterator' in jgrapht. Added an enum 'RandomWalkSearch' to select a random walk using the GraphSearch API. I am selecting the nodes at random and if a path exists, iterating through all possible paths before the node is found. Commit link for this: <https://github.com/Parth576/CSE-464-2023-prshah11/commit/a771e3df6d996fea9700418f0be7cbde8f93c120>
- I used the input2.dot uploaded on canvas to test the path for the random walk algorithm.

```
/home/parth/.jdk/openjdk-19.0.2/bin/java -javaagent:/ap
Graph successfully parsed!
Random Walk Iteration 1
a
a -> e
a -> e -> f
a -> e -> f -> h
Path not found

Random Walk Iteration 2
a
a -> e
a -> e -> g
a -> e -> g -> h
Path not found

Random Walk Iteration 3
a
a -> e
a -> e -> f
a -> e -> f -> h
Path not found

Random Walk Iteration 4
a
a -> e
a -> e -> g
a -> e -> g -> h
Path not found

Random Walk Iteration 5
a
a -> b
a -> b -> c
Path found!

Process finished with exit code 0
```

- Example usage:

```
GraphManager gm = new GraphManager();  
gm.importGraphFromDOT("src/input2.dot");  
GraphManager.Path result = gm.GraphSearch("a", "h",  
GraphManager.Algorithm.RandomWalkSearch);
```

## 5. Code review

- So, I pushed all my commits to the 'refactor' branch and created a pull request from refactor branch to master branch.
- <https://github.com/Parth576/CSE-464-2023-prshah11/pull/4>
- I put all the commit information in the pull request description and ensured that the tests were passing and there was no conflict between the branches. The link to the workflow run on the PR: <https://github.com/Parth576/CSE-464-2023-prshah11/actions/runs/4771697885/jobs/8483785938>
- The PR was merged successfully <https://github.com/Parth576/CSE-464-2023-prshah11/commit/97243698f3b07697d08e6bc2e2c4811e2fef5193>

